TECHNIQUES FOR DYNAMICALLY LOADING MODULES FOR DEVICES DISCOVERED IN A STORAGE NETWORK

CROSS-REFERENCES TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional Patent Application No. 60/228,546 entitled "STORAGE NETWORK CONTROL PROGRAM ARCHITECTURE FOR DYNAMIC INCORPORATION OF DEVICE DEPENDENT MODULES FOR DEVICES DISCOVERED DURING A NETWORK SCAN" filed August 28, 2000, the entire disclosure of which is herein incorporated by reference for all purposes.

BACKGROUND OF THE INVENTION

The present invention relates to storage networks and more specifically to techniques for facilitating dynamic loading of code modules in storage network application programs.

Storage area networks (or SANs) provide a flexible and scalable infrastructure for data storage that is capable of delivering any-to-any connectivity between host systems and storage devices. SANs offer several benefits over traditional server-centric storage models including improved scalability of storage resources, availability, data protection, centralized storage management, improved data access speeds, and others. SANs allow enterprises to meet the constantly changing data storage requirements of today's large and rapidly growing networks, and help relieve storage-related bottlenecks typically associated with traditional server-centric storage techniques. As a result, SANs are increasingly replacing or supplementing traditional server-centric storage implementations.

The emergence of SANs has created the need for new storage management tools and applications which perform functions such as storage partitioning, storage provisioning, allocation of storage to hosts, data replications, backup services, monitoring status of SAN devices, and other functions. In order to be effective, these applications/tools need to facilitate additions, deletions, and changes to SAN components/devices without any downtime. These SAN applications also need to facilitate software and hardware updates in a seamless manner. The applications need to be easily configurable and provide the flexibility to support new devices or newly configured SAN devices.

In light of the above, there is a need for techniques which allow SAN-related applications to facilitate changes to the SAN without having to restart the application or without having to power down the SAN.

BRIEF SUMMARY OF THE INVENTION

The present invention describes techniques for dynamically loading code modules in applications for monitoring and/or managing SANs without having to restart the application or power down the SANs. The code modules loaded by an application, while the application is executing, may be used by the application to perform a variety of SAN related functions including automated discovery and visualization, device management, performance and status monitoring, and other like functions. Since these tasks can be performed without requiring down time of the application or of the SAN itself, techniques according to the present invention increase the efficiency, level of flexibility, and ease of use of SAN applications, in addition to other benefits.

According to an embodiment of the present invention, techniques are provided for dynamically loading code modules in an application program executing on a data processing system coupled to a SAN. Upon execution, the program is configured to access device information, which may be stored in one or more files. The device information may comprise information identifying a set of SAN device identifiers and a set of code modules associated with the set of SAN device identifiers. The application program may be configured to load the set of code modules referenced by the device information into an address space of the executing application program. The loaded code modules may then be used by the application program to perform SAN related functions. While the program is executing, a signal may be provided to the application program indicating that the device information has been modified. In response to the signal, the application program maybe configured to delete one or more code modules referenced by the device information before modification from the address space of the executing program, access the modified device information, and load the code modules referenced by the modified device information into the address space of the executing program. The code modules loaded in the address space of the application program may then be used to perform SAN related functions.

According to another embodiment of the present invention, an application program executing on a data processing system coupled to a SAN is configured to access device information wherein the device information includes information related to a set of SAN device identifiers and information identifying a set of code modules associated with the

set of SAN device identifiers, the device information including information related to a first SAN device identifier and a first code module associated with the first SAN device identifier. The set of code modules identified in the device information are loaded into an address space of the executing program. The loaded set of code modules may be used to monitor devices coupled to the SAN whose device identifiers match identifiers in the set of SAN device identifiers. While the program is executing, a signal may be provided to the program indicating that the device information has been modified such that the information related to the first SAN device identifier has been removed/deleted from the device information. In response to the signal, the present invention is configured to delete the first code module associated with the first SAN device identifier from the address space of the executing program.

According to yet another embodiment of the present invention, an application program executing on a data processing system coupled to a SAN is configured to access information related to a first SAN device identifier wherein the accessed information includes information identifying a first code module associated with the first SAN device identifier. The embodiment of the present invention may load the first code module into an address space of the executing program. According to an embodiment of the present invention, the first code module may then be used to manage/monitor devices coupled to SAN whose device identifiers match the first SAN device identifier. While the program is executing, the program may receive a signal indicating that the information related to the first SAN device identifier has been modified, such that a second code module is associated with the first SAN device identifier instead of the first code module. In response to the signal, the present invention may delete the first code module associated with the first SAN device identifier from the address space of the executing program, and load the second code module into the address space of the executing program. According to an embodiment of the present invention, the second code module may then be used to manage/monitor devices coupled to SAN whose device identifiers match the first SAN device identifier.

According to an embodiment of the present invention, techniques are provided for dynamically loading code modules in an application program executing on a data processing system coupled to a SAN. In this embodiment, the program may be configured to access devices information comprising a set of SAN device identifiers including a first SAN device identifier, and information identifying code modules associated with SAN device identifiers in the set of SAN device identifiers. The devices information includes information identifying a first code module associated with the first SAN device identifier. The program

may load the set of code modules associated with the set of SAN device identifiers, including the first code module, into an address space of the executing program. During execution of the program, the program may receive a signal indicating that the devices information has been modified such that the modified devices information includes a second SAN device identifier and a second code module associated with the second SAN device identifier, the second SAN device identifier not included in the set of SAN device identifiers included in the devices information before modification. In response to the signal, the program, according to an embodiment of the present invention, may load the second code module into the address space of the executing program. The second code module and the previously loaded code modules may then be used to monitor/manage devices coupled to the SAN.

According to yet another embodiment of the present invention, an application program executing on a data processing system coupled to a SAN is configured to access information related to a SAN device identifier, the information including information identifying a code module associated with the SAN device identifier. The program may load the code module into an address space of the executing program. While the program is executing, it may receive a signal indicating that the code module has been modified. In response to the signal, according to the teachings of the present invention, the program may delete the previously loaded code module from the address space of the executing program, and load the modified code module into the address space of the executing program. The modified code module, instead of the pre-modified code module, may then be used to monitor/manage SAN devices having device identifiers which match the SAN device identifier.

The foregoing, together with other features, embodiments, and advantages of the present invention, will become apparent when referring to the following specification, claims, and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a simplified block diagram of a computer network which might incorporate an embodiment of the present invention;

Fig. 2 is a simplified block diagram of a computer system according to an embodiment of the present invention;

Fig. 3 is a simplified high-level flowchart depicting processing performed according to an embodiment of the present invention;

Fig. 4 is a simplified high-level flowchart depicting processing which may be performed in step 308 of Fig. 3 by an embodiment of the present invention to perform SRM related tasks; and

Fig. 5 is a simplified high-level flowchart depicting processing which may be performed in step 320 of Fig. 3 by an embodiment of the present invention to perform SRM related tasks.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides techniques which enable applications executing on a computer system coupled to SANs to dynamically drop/load code modules. According to an embodiment of the present invention, the dynamic loading of modules can be performed without experiencing any down time for the application or without having to restart the application or without having to power down the SAN. Fig. 1 is a simplified block diagram of a computer network 100 which might incorporate an embodiment of the present invention. As depicted in Fig. 1, computer network 100 comprisesa communication network 104, a SAN 130 which coupled to communication network 104, and one or more computers 102 which are coupled to communication network 104 or are part of the SAN itself. Distributed computer network 100 depicted in Fig. 1 is merely illustrative of an embodiment incorporating the present invention and does not limit the scope of the invention as recited in the claims. One of ordinary skill in the art would recognize other variations, modifications, and alternatives. For example, a plurality of SANs may be coupled to communication network 104.

Communication network 104 provides a mechanism for allowing communication and exchange of information between the various computer systems coupled to communication network 104 (e.g., computer systems 102-a, and 102-b) and SAN 130. Communication network 104 may itself be comprised of many interconnected computer systems and communication links. While in one embodiment, communication network 104 is the Internet, in other embodiments, communication network 104 may be any suitable communication network including a local area network (LAN), a wide area network (WAN), a wireless network, a intranet, private networks, public networks, switched networks, and the like.

Several different types of communication links may enable the connections between the various entities depicted in Fig. 1. The communication links may be hardwire links, optical links, satellite or other wireless communications links, wave propagation links,

or any other mechanisms for communication of information. Various communication protocols may be used to facilitate communication between the various systems shown in Fig. 1. These communication protocols may include TCP/IP, HTTP protocols, extensible markup language (XML), wireless application protocol (WAP), IETF, vendor-specific protocols, customized protocols, Fibre Channel protocols, and others.

SAN 130 may comprise several heterogenous, multi-vendor components or devices such as switches, hubs, routers, storage subsystems, cables, connectors, computer systems, and the like. For purposes of this application, the word "device" is used interchangeably with the word "component". A SAN generally comprises of four classes of components or devices: (1) end-user platforms such as computer systems 102, thin clients, etc.; (2) server systems such as servers 108 depicted in Fig. 1; (3) storage devices and storage subsystems 106 which are used to store data/information; and (4) interconnect entities which provide interconnectivity between the various SAN components and with other devices/networks. Interconnections between the various SAN components may be provided by connectivity technologies such as LAN technologies, WAN technologies, Fibre Channel network technologies, and the like.

Fig. 1 depicts a SAN 130 based upon Fibre Channel technology (sometimes referred to as a "Fibre Channel SAN"). As shown in Fig. 1, Fibre Channel SAN 130 comprises a plurality of components such as storage devices or storage subsystems 106, server systems 108, and end-user platforms such as computer system 102-c coupled to a Fibre Channel network 110. The devices may be connected to Fibre Channel network 110 using Fibre Channel adapters. Storage devices 106 may include tapes, storage disks, optical storage devices, RAID devices, tape libraries, and other types of storage devices. Fibre Channel network 110 may itself be composed of various types of interconnect entities such as switches, hubs, routers, bridges, and the like. Fibre Channel network 110 provides an infrastructure for high performance, any-to-any interconnect for server-to-server or server-to-storage traffic. Fiber Channel allows multiple protocols for interconnecting the various components/devices over a single infrastructure. It should be apparent that SAN 130 depicted in Fig. 1 is merely illustrative of an embodiment incorporating the present invention and does not limit the scope of the invention as recited in the claims. One of ordinary skill in the art would recognize other variations, modifications, and alternatives.

Due to the complexity and heterogenous nature of a SAN, in order to be effective and useful, it is important that SAN applications be able to facilitate additions/removals of SAN devices, facilitate changes to SAN devices, handle updates to

SAN-associated hardware, firmware, and software (e.g. support for new features devices, operating systems, and protocols) dynamically without having to power down the SAN or without having to restart the SAN application. According to the present invention, techniques are provided which use plug-in architecture models to facilitate addition/removal of SAN devices and facilitate changes to hardware/software related to the SAN in a dynamic manner. According to an embodiment of the present invention, techniques are provided for dynamically loading/dropping code modules associated with SAN devices/components while the SAN application program is executing on a computer system.

According to an embodiment of the present invention, the functions/tasks performed by the present invention may be implemented in software modules which may be executed by a data processing system such as computer system 102. The software modules implementing the features of the present invention may be incorporated into a SAN application program or alternatively may be used by SAN application programs. The software modules may also be implemented as a stand alone program offering services to other SAN application programs. The software modules may be executed by computer systems which are part of the SAN (e.g. computer system 102-c depicted in Fig. 1) or by computer systems which may be remotely coupled to the SAN (e.g. computer systems 102-a and 102-b which are coupled to SAN 130 via communication network 104).

Fig. 2 is a simplified block diagram of a computer system 102 according to an embodiment of the present invention. As shown in Fig. 2, computer system 102 may include at least one processor 204, which communicates with a number of peripheral devices via bus subsystem 202. These peripheral devices may include a storage subsystem 212, comprising a memory subsystem 214 and a file storage subsystem 220, user interface input devices 210, user interface output devices 208, and a network interface subsystem 206. The input and output devices allow user interaction with computer system 102. A user may be a human user (e.g., a SAN system administrator), a device, a process, another computer, and the like. Network interface subsystem 206 provides an interface to outside networks, including an interface to communication network 104 and/or Fibre Channel network 110, and may be coupled via the network to corresponding interface devices in other computer systems.

User interface input devices 210 may include a keyboard, pointing devices such as a mouse, trackball, touchpad, or graphics tablet, a scanner, a barcode scanner, a touchscreen incorporated into the display, audio input devices such as voice recognition systems, microphones, and other types of input devices. In general,use of the term "input

device" is intended to include all possible types of devices and ways to input information into computer system 102 or to communication networks coupled to computer system 102.

User interface output devices 208 may include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices. The display subsystem may be a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), or a projection device. The display subsystem may also provide non-visual display such as via audio output devices. In general, use of the term "output device" is intended to include all possible types of devices and ways to output information from computer system 102.

Storage subsystem 212 may be configured to store the basic programming and data constructs that provide the functionality of the computer system and of the present invention. For example, according to an embodiment of the present invention, software modules implementing the functionality of the present invention may be stored in storage subsystem 212. These software modules may be executed by processor(s) 204 of computer system 102. In a distributed environment, the software modules may be stored on a plurality of computer systems and executed by processors of the plurality of computer systems. Storage subsystem 212 may also provide a repository for storing various databases which may be used to store information according to the teachings of the present invention. Storage subsystem 212 may comprise memory subsystem 214 and file storage subsystem 220.

Memory subsystem 214 may include a number of memories including a main random access memory (RAM) 218 for storage of instructions and data during program execution and a read only memory (ROM) 216 in which fixed instructions are stored. File storage subsystem 220 provides persistent (non-volatile) storage for program and data files, and may include a hard disk drive, a floppy disk drive along with associated removable media, a Compact Digital Read Only Memory (CD-ROM) drive, an optical drive, removable media cartridges, and other like storage media. One or more of the drives may be located at remote locations on other connected computers at another site coupled to communication network 104 or Fibre Channel network 110. Information stored according to the teachings of the present invention may also be stored by file storage subsystem 220.

Bus subsystem 202 provides a mechanism for letting the various components and subsystems of computer system 102 communicate with each other as intended. The various subsystems and components of computer system 102 need not be at the same physical location but may be distributed at various locations within distributed network 100. Although bus subsystem 202 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple busses.

Computer system 102 itself can be of varying types including a personal computer, a portable computer, a workstation, a computer terminal, a network computer, a mainframe, a kiosk, a PDA, a communication device such as a cell phone, or any other data processing system. Due to the ever-changing nature of computers and networks, the description of computer system 102 depicted in Fig. 2 is intended only as a specific example for purposes of illustrating the preferred embodiment of the computer system. Many other configurations of a computer system are possible having more or fewer components than the computer system depicted in Fig. 2.

Fig. 3 is a simplified high-level flowchart 300 depicting processing performed according to an embodiment of the present invention. As described above, the software modules implementing the present invention may be incorporated into a SAN application program, or may be implemented as a stand alone program(s) offering services to other SAN application programs. Flowchart 300 and the related description describes an embodiment wherein the software modules are assumed to be incorporated into the SAN application. Accordingly, execution of the SAN application invokes services provided by the present invention.

As depicted in Fig. 3, a SAN application program (incorporating software modules implementing features of the present invention) may be executed on a computer system which may be either locally or remotely coupled to a SAN such as SAN 130 depicted in Fig. 1 (step 302). Examples of SAN application programs include a SAN management program, a program for visualizing the SAN layout, and the like.

Upon execution, the SAN application program may access and read information related to SAN devices supported by the SAN application program (step 304). Examples of SAN devices which may be supported by the application program may include various types of switches, hubs, routers, storage subsystems, cables, connectors, computer systems, and the like, manufactured by one or more vendors. According to an embodiment of the present invention, the SAN devices information accessed by the application program in step 304 contains information identifying one or more SAN devices supported by the application program. For each SAN device supported by the SAN application program, the SAN devices information may comprise information identifying a unique identifier associated with the SAN device and which uniquely identifies the SAN device. For example, if Simple Network Management Protocol (SNMP) protocol is used to perform SAN management, the unique identifiers may be "sysOIDs" (system object identifiers) used by the

SNMP protocol to identify the SAN devices. It should be apparent that other techniques for identifying SAN devices may also be used within the scope of the present invention.

In addition to the unique identifiers, for each SAN device, the SAN devices information may also include information indicating the location, name, and/or version of one or more code modules associated with the particular SAN device. For example, for a particular SAN device, the SAN devices information may identify the pathname of a directory storing a code module associated with the particular SAN device and the name (e.g., filename) of the code module. A version of the code module may also be indicated. According to an embodiment of the present invention, Uniform Resource Locators (URLs) may be used to identify names of code modules associated with the SAN device and the locations from where the code modules may be accessed. The SAN devices information accessed in step 304 may also comprise other information related to the SAN devices, such as a textual description of the SAN devices, and the like. The term "code module" as used in this application is meant to refer to one or more software code modules, scripts, patches, programming data or any other data.

The SAN devices information may be stored in a file which may be accessed and read by the application program in step 304. Alternatively, the SAN devices information may be distributed across a plurality of files. According to an embodiment of the present invention, the SAN devices information is stored in a plurality of files with each file storing device information for a single SAN device. These files may be referred to as "property files" with each property file storing device information for a particular SAN device. The one or more files storing the SAN devices information may be stored either locally or remotely from the computer system executing the application program.

Referring back to Fig. 3, the application program may then load the code modules referenced by the SAN devices information (read in step 304) into the application space (or address space or virtual space) of the SAN application program (step 306). As discussed above, according to an embodiment of the present invention, the SAN devices information comprises URLs indicating the location and name of code modules associated with SAN devices supported by the application program. In this embodiment, the application program uses the URLs to access, read, and load code modules identified by the URLs. According to an embodiment of the present invention, a URL Class Loader may be used to load the code modules. A URL Class Loader takes a URL as an input parameter and loads the code module identified by the input parameter URL. It should be apparent that various

other techniques known to those of ordinary skill in the art may also be used to load code modules in alternative embodiments of the present invention.

The code modules loaded in step 306 may then be used by the SAN application program to perform various SAN related functions and/or tasks (step 308). For example, the application program may use the code modules to monitor the SAN, to manage SAN devices coupled to the SAN, to perform configuration management on SAN devices, and to perform other like functions related to the SAN.

According to the teachings of the present invention, while executing on the computer system, the application program may receive a signal indicating that the SAN devices information has been modified or that one or more code modules referenced by the previously accessed SAN devices information have been modified (step 310). The SAN devices information is considered modified when information is added to or deleted from the SAN devices information, and/or when changes are made to the SAN devices information.

The SAN devices information may have been modified for various reasons. For example, the SAN devices information may have been modified to add support for one or more new SAN devices which have been added or will be added to the SAN. Thismay be accomplished by adding information related to new SAN device(s) to the SAN devices information. For example, in the embodiment of the present invention where the SAN devices information is stored in a plurality of files with each file (property file) storing device information for a particular SAN device, a SAN system administrator may modify the SAN devices information by adding a new property file containing information for each new device to be added to the SAN and to be supported by the application program.

The SAN devices information may also have been modified to remove/drop support for one or more SAN devices. This may be accomplished by removing or deleting device information corresponding to the "dropped" SAN devices from the SAN devices information. For example, in the embodiment of the present invention where SAN devices information is stored in a plurality of files with each file (property file) storing device information for a particular SAN device, a SAN system administrator may remove/drop support for one or more previously supported SAN devices by deleting property files corresponding to the SAN devices to be dropped.

The SAN devices information may also have been modified to modify/update the name and/or location of one or more code modules referenced by the SAN devices information. For example, if a new version of a code module is available for a particular SAN device, the code module name and location information associated with the particular

SAN device in the SAN devices information may be changed to refer to the new version of the code module. The code module name and location information may also be modified to refer to another, possibly older, version of the code module (e.g. if the code module presently referenced by the SAN devices information was determined to contain excessive defects/bugs, the SAN devices information may be changed to point to another less buggy version of the code module). It should be apparent that the SAN devices information may also be modified/amended for various other reasons not described above.

According to an embodiment of the present invention, the signal received in step 310 may specifically identify the modifications made to the SAN devices information. For example, the signal may contain information specifically identifying the property files which have been added or deleted or modified. Alternatively, in a simplified embodiment of the present invention, the signal received in step 310 may only indicate that the SAN device information has been modified without identifying the specifics of the modifications.

The signal received in step 310 may also contain information indicating that one or more code modules referenced by the previously accessed SAN device information have been modified (In this scenario, the SAN devices information may not have been modified.). A code module may have been modified for a variety of reasons. For example, a code module may have been modified to correct bugs or defects in the code module. A code module may also have been modified to add new features or to enhance the functionality of the code module (e.g. to add new features/functionality to the SAN device associated with the code module). A code module may also be modified for several other reasons.

According to an embodiment of the present invention, the signal received in step 310 may specifically identify the one or more code modules which have been modified. The signal may also identify the property files which reference the modified code modules. Alternatively, in a simple embodiment of the present invention, the signal received in step 310 may only indicate that at least one code module has been modified without providing any additional information specific to the modification/change.

The SAN application program then determines which of the previously loaded code modules are to be deleted from the address space of the application program in response to the signal received in step 310 (step 312). According to an embodiment of the present invention, the code modules to be deleted include code modules which have been identified as having been modified and/or deleted, and code modules corresponding to property files which have been identified as having being modified and/or deleted. For example, if the signal indicates that a particular property file has been modified, the application program

determines the one or more code modules corresponding to the modified property file and marks/tags those code modules for deletion from the address space of the application program. None of the previously loaded code modules may be marked for deletion if modification to the SAN devices information included only addition of new SAN devices (i.e. the signal received in step 310 indicates that the only change to the SAN devices information involved addition of one or more new property files corresponding to newly supported SAN devices). In embodiments or the present invention wherein the signal received in step 310 does not specifically identify the modified files or code modules, the application program may mark all the previously loaded code modules for deletion from the address space of the application program.

The SAN application program then deletes the code modules marked/tagged for deletion in step 312 from its address space (step 314).

The SAN application program then determines the code modules to be loaded into the address space of the application program in response to the signal received in step 310 (step 316). The application program determines which code modules are to be loaded based upon information included in the signal received in step 310. If the signal specifically identifies the code modules which have been modified, then those code modules are tagged to be loaded by the application program. If the signal identifies property files which have been modified or newly added, then code modules associated with those property files are marked for loading. In embodiments of the present invention where the signal received in step 310 does not specifically identify the code modules or the property files which have been modified, the application program may tag all the code modules referenced by the SAN devices information to be loaded.

The SAN application program then loads the code modules marked/tagged in step 316 into the address space of the application program (step 318). As described above in conjunction with step 306, various different techniques may be used to load the code modules into the address space of the application program. For example, in embodiments of the present invention where the SAN devices information comprises URL information, a URL Class Loader may be used to load the code modules identified by the URLs.

The SAN application program may then use the code modules loaded into the address space of the application program to perform various SAN related functions (step 320). The code modules loaded into the address space may include code modules loaded in step 318 in response to the signal received in step 310 and previously loaded codemodules, if

any, which were not deleted in step 314. Processing may then proceed with step 310 whenever the application program receives another signal according to step 310.

As described above, steps 310, 312, 314, 316, 318, and 320 are performed while the application program is executing on a computer system without having to restart the application program or without having to power down the SAN. Accordingly, the present invention facilitates addition, deletion, and movement of SAN devices in the SAN, addition of new features (e.g. addition of new discovery protocols, addition of features to the SAN devices, etc.), updates to software/hardware associated with the SAN devices, and other like functions in a dynamic manner without having to re-start the SAN application program or power down the SAN. This increases the efficiency, level of flexibility, and ease of use of SAN applications, in addition to other benefits.

As described above with respect to step 308 in Fig. 3, the code modules loaded into the address space of the application program may be used by the application program to perform a variety of SAN related functions. According to an embodiment of the present invention, the SAN application program may be a centralized storage resource management (SRM) application and may use the loaded code modules to manage the SAN and the storage related devices. Key SRM tasks include automated discovery and visualization, device management, performance and status monitoring, and other like tasks.

Fig. 4 is a simplified high-level flowchart 400 depicting processing which may be performed in step 308 of Fig. 3 by an embodiment of the present invention to perform SRM related tasks. As depicted in Fig. 4, the SAN application program may build a mapping (e.g. a hash map) between the unique SAN device identifiers referenced by the SAN devices information accessed in step 304 and corresponding code modules loaded into the address space of the application program in step 306 (step 402). For example, in the embodiment of the present invention where SAN devices information is stored in a plurality of files with each file (property file) storing device information for a particular SAN device, for each property file, the application program may build a mapping between the unique SAN device identifier contained in the property file and the code module (or modules) referenced by the property file and which has been loaded into the address space of the application program. In a SNMP protocol environment, "sysOIDs" correspond to the unique SAN device identifiers, and the mapping maps each sysOID to its corresponding loaded code module.

The SAN application program may then discover devices coupled to the SAN (step 404). During the discovery step, the application program determines the physical layout of the SAN. This may include determining SAN devices coupled to the SAN either locally or

remotely and the connections between the devices. Discovery may be performed using "out-of-band" communication techniques (e.g. SNMP over IP) or "in-band" methods" (e.g. directly over Fibre Channel). If an SNMP protocol is used, the SAN application program may query the devices for SNMP Management Information Base (MIB) data. The MIB data is then interpreted to find the information necessary to identify the topology of the SAN. Discovery of devices may be performed on a periodic basis.

As part of the discovery process according to step 404, for each discovered SAN device, the SAN application program determines a unique identifier associated with the SAN device and which uniquely identifies the SAN device. For example, the SNMP MIB data associates a "sysOID" with each device discovered as being coupled to the SAN.

For each device discovered in step 404, the application program determines if the discovered device is supported by the SAN application program (i.e. the application program determines if the unique device identifier associated with the discovered device is included in the mapping information generated in step 402), and if the discovered device is supported (i.e. the unique device identifier associated with the discovered device is included in the mapping information generated in step 402), the application program uses the mapping information to determine the code module (or modules) corresponding to the discovered device (step 406). For example, in an environment using the SNMP protocol, the application program determines if the sysOID associated with a discovered device is included in the mapping information (e.g. a hash map) generated in step 402. If the sysOID is included in the hash map, the application program determines the code module which is mapped to the particular sysOID and associates that code module with the discovered device.

For each SAN device discovered in step 404 and which is supported by the application program, the application program may then instantiate one or more objects for the SAN device using the code module associated with the device in step 406 (step 408). The objects instantiated in step 408 may then be used to monitor and/or manage the SAN devices and to perform other SRM functions related to the SAN devices (step 410).

Fig. 5 is a simplified high-level flowchart 500 depicting processing which may be performed in step 320 of Fig. 3 by an embodiment of the present invention to perform SRM related tasks. After the code modules have been loaded into the address space of the application program in step 318, the application program may update/modify the mapping information (e.g. the hash map) to reflect the new mappings between SAN device identifiers and code modules loaded into the address space of the application program. According to an embodiment of the present invention, the mapping information may be modified for only

those code modules which have been loaded in step 318. In alternative embodiments of the present invention, the application program may delete the old/previous mapping information and generate new mapping information to reflect the updated/modified devices information.

The SAN application program may then discover the devices coupled to the SAN (step 504). As described above, during the discovery step, the application program may determine the physical layout of the SAN by determining SAN devices coupled to the SAN either locally or remotely and the connections between the devices.

For each device discovered in step 504, the application program then determines if the discovered device is supported by the application program (i.e. the application program determines if the unique device identifier associated with the discovered device is included in the mapping information generated/modified in step 502), and if the discovered device is supported (i.e. the unique device identifier associated with the discovered device is included in the mapping information generated/modified in step 502), the application program uses the mapping information to determine the code module (or modules) corresponding to the discovered device. For example, in an environment using the SNMP protocol, the application program determines if the sysOID associated with a discovered device is included in the mapping information generated/modified in step 502, and if the sysOID is included in the mapping information, the application program determines the code module which is mapped to the particular sysOID and associates that code module with the discovered device.

For each SAN device discovered in step 504 and which is supported by the application program, the application program instantiates one or more objects for the SAN device using the code module associated with the device in step 506 (step 508). The objects instantiated in step 508 may then be used to monitor and/or manage the SAN devices and to perform other SRM functions related to the SAN devices (step 510). In this manner code modules are dynamically loaded and used by the application program to perform SRM related tasks without having to restart the application program or without having to power down the SAN.

Although specific embodiments of the invention have been described, various modifications, alterations, alternative constructions, and equivalents are also encompassed within the scope of the invention. The described invention is not restricted to operation within certain specific data processing environments, but is free to operate within a plurality of data processing environments. Additionally, although the present invention has been described using a particular series of transactions and steps, it should be apparent to those

skilled in the art that the scope of the present invention is not limited to the described series of transactions and steps.

Further, while the present invention has been described using a particular combination of hardware and software, it should be recognized that other combinations of hardware and software are also within the scope of the present invention. The present invention may be implemented only in hardware or only in software or using combinations thereof.

The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. It will, however, be evident that additions, subtractions, deletions, and other modifications and changes may be made thereunto without departing from the broader spirit and scope of the invention as set forth in the claims.